spimodfit
Explanatory Guide and Users Manual

H. Halloin[1]
MPE Garching
halloin@mpe.mpg.de

Software version 2.9
May 12, 2009

[1]Present address : APC, 11 place Marcelin Berthelot 75005 Paris, halloin@apc.univ-paris7.fr

Contents

1 Getting started **3**
- 1.1 Precompiled binaries . . . 3
- 1.2 Compiling from sources . . . 4
- 1.3 Software versions . . . 6
  - 1.3.1 Supported platforms . . . 6
- 1.4 Known issues . . . 6
- 1.5 Use and syntax of the parameter file . . . 7

2 Introduction **8**
- 2.1 Scope . . . 8

3 Method and Algorithms **10**
- 3.1 Imaging Data and Models . . . 10
- 3.2 Model Fitting . . . 11
  - 3.2.1 Approaches . . . 11
  - 3.2.2 Model Fitting: Details . . . 12

4 Using *spimodfit* **16**
- 4.1 General parameters . . . 16
- 4.2 Observation and IRF files . . . 17
  - 4.2.1 Loading and configuring the observation data . . . 17
    - 4.2.1.1 Path to the observation files . . . 18
    - 4.2.1.2 Background models selection and combination . . . 19
    - 4.2.1.3 Energy rebinning of the input data . . . 20
    - 4.2.1.4 Detector selection of the input data . . . 21
  - 4.2.2 Simulation of detector events . . . 21
  - 4.2.3 Rejection of bad data counts and background smoothing . . . 21
  - 4.2.4 Loading the IRF files . . . 22
- 4.3 Definition of the fitted components . . . 23
  - 4.3.1 Image components . . . 23
    - 4.3.1.1 Number and location of sky models . . . 23
    - 4.3.1.2 Sky models units and energy-dependent rescaling . . . 24
    - 4.3.1.3 Definition of detector and time intervals . . . 26

4.3.1.4 Default parameters values and fitting flags 27
4.3.1.5 Configuration of the output sky models 30
4.3.2 Point sources components 31
4.3.3 Source flux 31
Index of *spimodfit* parameters 32

Chapter 1

Getting started

Either you are using a binary version of *spimodfit* or want compile it from sources, the OSA environment has to be set. At MPE, this is done by sourcing the configuration script locating in the OSA subdirectory :

```
# source <Path_to_osa>/init_osa best
```

`<Path_to_osa>` stands for `/afs/ipp-garching.mpg.de/mpe/gamma/instruments/integral/isdc/osa` at the MPE. In the above command line, # stands for the UNIX or LINUX prompt (don't type it !).

Then, you have to set the `PFILES` environment variable to the directory of the parameter file. Usually, the parameter file is in the working directory, and so you should type :

```
# setenv PFILES .
```

1.1 Precompiled binaries

At the MPE, the documentation, parameter file and pre-compiled binaries of additional SPI software are in subdirectories of `/afs/ipp/mpe/gamma/instruments/integral/software`. So, in `spimodfit/2.9`, one can find :

- the program binaries in subdirectories whose name reflect the machine architecture and compiler. Their name is then of the form `SYSARCH_COMP`, where `SYSARCH` is the system architecture (as defined by the `SYS` environment variable, type `echo $SYS` to see its value), and `COMP` is the compiler (gcc, g++, icc, icpc, ...).
- the documentation directory (doc) that should, at least, contain a user manual called spimodfit.ps.
- an example of parameter file, called spimodfit.par.

If you are working on a machine for which a binary version exists, you probably have to use it. The next section (compiling from the source) is mostly intended for expert user in case of bug correction or compilation on a new system.

1.2 Compiling from sources

The source directory of additional SPI software can be found by following the `/afs/ipp/mpe/gamma/instruments/integral/software/halloin_src` symbolic link. In the `spimodfit/2.9` subdirectory, the following items should be present :

- source (*.cpp, *.c) and header (*.h) C++ and/or C files
- an `ac_stuff` subdirectory containing a slightly modified (in order to support 64 bits machine) configuration scripts from ISDC.
- a `doc` containing the software documentation
- a `makeisdc.in` file, used to configure the installation process.

This version of *spimodfit* depends on some locally installed libraries, listed in table 1.2. These libraries should have been compiled on the same architecture and with the same compiler. These libraries are located in subdirectories of `/afs/ipp/mpe/gamma/instruments/integral/software/halloin_src`.

Name	Version	subdirectory	Comments
dal	OSA 5.1	`CorrectedDAL/dal`	recompiled version (corrected for use with a 64 bits machine)
dal3gen	OSA 5.1	`CorrectedDAL/dal3gen`	recompiled version (corrected for use with a 64 bits machine)
CommonCLib	2.1	`CommonCLib/2.1`	OSA-independent C libraries for general use
spicommonlib	2.3	`spicommonlib/2.2`	collection of common C++ utilities, objects and methods used for local SPI programs.
fftw3	3.0.1	`fftw-3.0.1`	The "Fastest Fourier Transform In The West" package. Not directly used in *spimodfit* but necessary to some subroutines of `spicommonlib`.

Table 1.1: Required local libraries for compilation of *spimodfit* .

Assuming that all required libraries are available for the appropriate machine and compiler version, the compilation procedure is the following :

1. Clean any previous installation :

   ```
   # make distclean
   ```

   Please note that you may have to use gmake instead of make on some systems.

2. Configure the current installation :

```
# ./ac_stuff/configure
```

This configure script accepts many parameters and optional configuration flags. You may have an overlook of these option by typing `# ./ac_stuff/configure --help`. In particular, the compiler name and options can be set manually on the configure line by typing :

```
# ./ac_stuff/configure CC=... CFLAGS=... CXX=... CXXFLAGS=...,
```

where `CC` is the name of the C compiler and `CFLAGS` is the list of options to pas to the C compiler. `CXX` and `CXXFLAGS` follow the same definition for the C++ compiler. At the MPE, a compilation with the Intel compiler and a optimized code for Pentium 4 an compatible CPUs would then be :

```
# ./ac_stuff/configure CC=icc CFLAGS=-axW CXX=icpc CXXFLAGS=-axW,
```

The program will anyhow be compiled with the `-O3` and `-Wno-deprecated` options.

3. Build the binary :

   ```
   # make all
   ```

   This should compile the program and put it in a subdirectory called `SYSARCH_COMP`, where `SYSARCH` is the system architecture (as defined by the `SYS` environment variable, type `echo $SYS` to see its value), and `COMP` is the compiler (gcc, g++, icc, icpc, ...).

What to do if anything went wrong in one of the previous step ? Well ... First look carefully at the error message(s) to see if it is not an OSA or local system configuration problem. Second, if it seems to be a bug in any of the configuration script or even in the source code, you may consider to one (or maybe much more ?) cup of coffee and dive into the code and scripts. In desperate cases, you may also contact the programmer to complain about the software (eventually, (s)he will be able to identify and maybe solve your problem...).

1.3 Software versions

Date	Version	Comments
?? ?? 2005	1.0	(H.Halloin) First separate version from *spidiffit* version 26, various bug fixes, improve variability handling, allow zero source and/or skymap.
22 Feb 2005	2.0	(H.Halloin) Remove NAG subroutines : use LAPACK (eigen vectors) + lbfgs_bcm (minimization) + tn (minimization), correct some minor bugs.
26 Feb 2005	2.1	(H.Halloin) Remove eigenvalues calculation and replace with direct hessian (6-7 times faster).
02 Mar 2005	2.2	(H.Halloin) Add possibilities for detector selection on entry
12 Mar 2005	2.3	(H.Halloin) Output sources catalogue
13 Mar 2005	2.4	(H.Halloin) Possibility to load default parameters from a previous run.
21 Mar 2005	2.5	(H.Halloin) Add Levenberg-Marquadt algorithm.
24 Mar 2005	2.6	(H.Halloin) Modify the way to specify the energy modulation.
19 Apr 2005	2.7	(H.Halloin) Use common libraries instead of locally defined
20 May 2005	2.8	(H.Halloin) Allow more flexibility in time variation (combination of multiple definitions), first documentation release (Aug 2006)

1.3.1 Supported platforms

The following table gives some information about the platforms on which *spimodfit* has been compiled :

- System : value of the `SYS` environment variable
- Platform : hardware architecture, as given by `uname -i`
- Kernel version : as given by `uname -rs`
- Operating system : as given by `uname -o`

Vers.	OSA	system	platform	kernel	OS	compiler
2.8	5.1/linux	i386_rh90	i386	Linux 2.4.21-241-smp	GNU/Linux	g++/gcc icpc/icc

1.4 Known issues

- *spimodfit* handles time variability through the use of splines. The spline order can be 0 to 5, 0 corresponding to a piecewise constant function (with one scaling parameter per interval) and 3 corresponding to cubic splines. In many cases, when using n-order splines ($n \geq 1$), the fitting algorithm fails to find the optimal parameters. This is thought to be due to over-parametrized time variability because of the additional parameters of the splines.

1.5 Use and syntax of the parameter file

The parameters of the the program are defined in a file called spimodfit.par. This file should be readable in one of the directory set in the `PFILES` environment variable. Usually, the user updates the parameter file in the working directory, so one should set the variable accordingly : `setenv PFILES ..`

The syntax of the file follow the IRAF standards (that is the one used by ISDC softwares).

In this file, each parameter is described in the format :

`parameter_name, parameter_type, parameter_mode, default_value, minimum_value, maximum_value, description_prompt`

The parameter_type can be "b" (boolean), "i" (integer), "r" (real), "s" (string) or "f" (filename). The "f" type can be followed by any combination of "r" (read access), "w" (write access), "e" (existence of file) "n" (absence of file). Thus, the "fw" type means to test whether the file given as a value of the parameter is writable.

The parameter_mode can be "a" (auto), "h" (hidden), "q" (query), "hl" (hidden+learn), "ql" (query+learn). With the "a" mode, the effective mode is set to the value of the parameter named as "mode" in the parameter file. If the "mode" parameter is not found, the effective mode is set to "h". With the "h" mode, no questions are asked, unless the default value is invalid for a given data type, whereas with the "q" mode you are always asked for a parameter. With the additional "l" mode, if an application changes a parameter value, the new value will be written to the parameter file when the application terminates.

spimodfit makes use of parameter types "i", "r" and "s".

Some of the parameters require to enter multiple arguments with the same parameters. This is done via a string which is then decomposed into different elements and interpreted. A multiple arguments parameter can be either a vector or a list :

- A vector is a array of values of the same kind (i.e. reals, integers, etc.), separated by blanks, for example : "1100.5 1433.0 1500.0" is a vector of 3 reals. This is a extension or the IRAF format coded in the OSA PIL library.

- A list is a array of values or fields (not necessary of the same kind) separated by commas. Sometimes an additional keyword can be put at the end of the list to modify its interpretation. For example : "1785-1802, 1802-1815, 1815-1826 keV" selects 3 energy ranges in keV. This possibility is a local extension of the IRAF standards.

Unless specified in the parameter description, all parameters are required to appear in the parameter file, even if their value is not used by the program.

Blank lines and lines beginning with '#' (comments) are ignored.

Chapter 2

Introduction

spimodfit is intented to perform "model fitting" with SPI data. By "model fitting", we mean the fit of a linear combination of sky emmission models, point sources and background. This program is based on a former model fitting program for SPI, *spidiffit* developped by A. Strong. The main changes in *spimodfit* (w.r.t *spidiffit*) are :

- a more flexible handling of time variability (for all kind of sources)
- time-dependent instrument response function (IRF)
- the use of NAG-free constrained minimization procedures and the addition of a Levenberg-marquadt algorithm
- optimized matrix calculations and memory usage
- more output files (updated catalog of sources, background model, residues, etc.)

spimodfit is mainly dedicated to the study of long duration observations, large scale emission, focusing on the emission morphology (as opposed to sopectral tools like XPEC).

The aim and basic algorithm of *spimodfit* are anyhow identical to those of *spidiffit*. The next introductive sections are then directly taken from the *spidiffit* user manual (2^{nd} version).

2.1 Scope

The coded-mask imaging γ-ray spectrometer SPI on the INTEGRAL space observatory will detect point sources and diffuse extended emission with an angular accuracy of about 1° over its energy range of 40–8000 keV. The purpose of the software package described here is to represent the measurement in terms of spatial models on the sky, fitting parameters of these models and estimating their uncertainty. This tool is oriented towards large-scale surveys (eg. GCDE), which combine a large set of individual pointings of the spacecraft. It concentrates on fitting spatial as opposed to spectral models, although (through using the different energy channels of the measurement) it will be an important method to generate spectra of diffuse emission.

The principal use will be fitting to maps of physically-based tracers of γ-ray emission. Examples are fitting the 1809 keV ^{26}Al line to free-free or IR emission maps, fitting diffuse continuum

γ-ray s to gas (HI, CO) maps. It will also allow fitting spatial functions (e.g. Gaussian, exponential) where there is no 'physical' model available. Fitting components is the best method to get the spectra of diffuse emission when a good spatial (but a poor spectral) model is available. The analysis of measurements with this tool is complementary to deriving results from deconvolved images (e.g. from *spiskymax*) and methods specifically designed for point sources (e.g. *spiros*).

The algorithm performs fitting of raw data (binned counts for many observations) to multi-component models using the full instrument response information. In addition to parameter estimation, a Bayesian statistical analysis is used to include information on model characteristics and other prior information, so that the uncertainties of the fitted parameters are properly assessed. The prime results of the package are fitted model parameters with their uncertainty; complementing these, the fitted models are given also in the forms of skymaps and spectra.

The package will be referred to as *spimodfit* . The package is closely related to imaging packages: response, convolution, background treatments are common or similar.

Chapter 3

Method and Algorithms

This chapter is entirely taken from the *spidiffit* user manual version 2. It describes the overall data representation and MCMC fitting method. *spimodfit* can also fit the parameters from more 'classical' first or second derivatives methods (modified and truncated newton, Levenberg-Marquadt). These algorithms are not (yet) documented though their use is explained in a latter section.

3.1 Imaging Data and Models

We distinguish image space and data space in the usual way, and define the instrument response as the relation between them. The image is I_j and the expected data is d_k. The expected background is b_k Let R_{jk} be the response of data element k to image element j. Then

$$d_k = \sum_j R_{jk} I_j + b_k$$

An *image model* is a parameterized algorithm for composing an image from components. For a linear model,

$$I_j = \sum_i \theta_i M_{ij}$$

where θ_i are the model parameters.

More generally, the image will still be described by a sum of components, but the image components will be non-linear functions of the parameters (e.g. Gaussian, exponential) and each component is described by several parameters $\overline{\theta_i}$:

$$I_j = \sum_i M_{ij}(\overline{\theta_i})$$

Similarly the background can be constructed from components of a background model B_{ik}

$$b_k = \sum_i \theta_i B_{ik}$$

where θ_i now introduces background parameters. The sums in the above expressions are over the appropriate subsets of parameters for image and background model respectively. In this way we can treat image and background model in the same way in the subsequent analysis, and θ_i includes both. The only formal difference between image and background model is that the image is convolved with R_{jk} and the background is not:

$$d_k = \sum_j R_{jk} \sum_{i=1}^{i=N_I} \theta_i M_{ij} + \sum_{i=N_I+1}^{i=N_I+N_B} \theta_i B_{ik}$$

where there are N_I image components and N_B background components, and $N = N_I + N_B$.

In our modelling approach, the measured signal is represented through model components:

$$d_k = \sum_i \theta_i S_{ik}$$

where the sky part is:

$$S_{ik} = \sum_j R_{jk} M_{ij}, \qquad i = 1, N_I$$

and the instrumental-background part is:

$$S_{ik} = B_{ik}, \qquad i = N_I + 1, N_I + N_B$$

In the non-linear case we have to convolve explicitly for each parameter set (i.e. we cannot pre-convolve to get S_{ik}):

$$d_k = \sum_j R_{jk} \sum_{i=1}^{i=N_I} M_{ij}(\overline{\theta_i}) + \sum_{i=N_I+1}^{i=N_I+N_B} \theta_i B_{ik}$$

The procedure above is for a single energy and hence a implies a diagonal response in energy space. However it is simple to generalize to the case of a dataset including many energy channels and parameters for each energy, so that the solution constitutes an *energy deconvolution*. In this case S_{ik} includes the off-diagonal response terms. From the point of view of the analysis technique there is no difference, just S_{ik} is larger (by a factor equal to the square of the number of energy channels), and d_k and θ_i are correspondingly expanded.

3.2 Model Fitting

3.2.1 Approaches

Various approaches are possible based on well-established principles. An important point is that the data are Poisson so that any method must be able to handle such data correctly.

For COMPTEL, EGRET and other γ-ray missions the maximum-likelihood-ratio method has been widely used. The properties of this method and its used for the generation of error-estimates are well understood. It is a classical statistical method.

In recent years much attention has been paid to Bayesian methods, and their advantages are well documented (e.g. Loredo, vanDyk). Bayesian approaches have been incorporated in e.g. Chandra standard packages. Especially in GRB and cosmological statistical analyses it has become the 'method of choice' in many publications (see reference list). The main advantage comes in handling multi-parameter problems where the classical methods have fundamental limitations. In particular the handling of so-called 'nuisance parameters' is in general not possible in classical methods but straightforward in the Bayesian approach. In view of the large amount of literature on this topic explicitly for astronomical applications, and the availability of algorithms (and also software), it is proposed to take advantage of this in providing such an approach for *spimodfit* . Up to a few years ago the method was regarded as impractical for many-parameter problems because of the difficulty of evaluating multi-dimensional integrals, but now methods are readily available (Markov Chain Monte Carlo) which make this both tractable and relatively straightforward to implement.

Last but not least one of the leading Bayesian groups is at the Centre for Interdisciplinary Studies in Garching (neighbouring institute to MPE)

Possibily both approaches should be developed.

3.2.2 Model Fitting: Details

The objective of the model fitting analysis is now to extract information about θ_i in the form of posterior probability distributions, and their moments (mean, standard deviation etc) and any other functions of interest (e.g. the total image).

Fitting Constraint

The likelihood function is:

$$L(D|\bar{\theta}) = \prod_k e^{-d_k} d_k^{n_k}/n_k!$$

where n_k are the measured data (denoted collectively by D).

and the posterior probability $P(\bar{\theta}|D)$ is expressed in terms of the likelihood and the prior probability $Pr(\bar{\theta})$ using Bayes theorem:

$$P(\bar{\theta}|D) = L(D|\bar{\theta})Pr(\bar{\theta})/P(D)$$

where P(D) is known as the *evidence*.

Suitable analytical forms for the prior $Pr(\bar{\theta})$ are given by e.g. van Dyk et al. (2001); these can be used to incorporate the user's knowledge into the problem.

Results

The posterior for one parameter is obtained by marginalizing over the other parameters:

$$P(\theta_i|D) = \int_{i' \neq i} P(\bar{\theta}'_i|D) d^N\theta'$$

and its mean value is

$$< \theta_i|D >= \int \theta_i P(\theta_i|D) d\theta_i = \int \theta_i P(\bar{\theta}|D) d^N\theta$$

with standard deviation

$$\Delta\theta_i|D = sqrt \int (\theta_i - < \theta_i|D >)^2 P(\theta_i|D) d\theta_i = sqrt \int (\theta_i - < \theta_i|D >)^2 P(\bar{\theta}|D) d^N\theta$$

The results are expressed in terms of the posteriors and mean and standard deviations of the parameters, but also more conveniently in terms of the fitted skymaps, i.e. the input maps multiplied by the fitted parameters together with the error estimates.

$$< I_{ij} >=< \theta_i|D > M_{ij}$$
$$\Delta I_{ij} = (\Delta\theta_i|D) M_{ij}$$

The average total map is the sum of the average maps:

$$I_j = \sum_i \theta_i M_{ij}$$

$$< I_j >= \int \sum_i \theta_i M_{ij} P(\bar{\theta}|D) d^N\theta = \sum_i < \theta_i|D > M_{ij}$$

but the error is more complicated to compute since it involves the correlations between the θ_i:

$$\Delta I_j = sqrt \int (I_j - < I_j >)^2 P(\bar{\theta}|D) d^N\theta$$

$$\Delta I_j = sqrt \int (\sum_i \theta_i M_{ij} - \sum_i < \theta_i|D > M_{ij})^2 P(\bar{\theta}|D) d^N\theta$$

$$\Delta I_j = sqrt \int (\sum_i [\theta_i M_{ij} - < \theta_i|D > M_{ij}])^2 P(\bar{\theta}|D) d^N\theta$$

In general we want statistics for a linear function of the parameters:

$$f = \sum w_i \theta_i$$

The full posterior distribution for a given set of w_i must come from the MCMC method (see below) but the standard deviation can be more generally derived from the covariances as follows.

$$\Delta f = sqrt \int (\sum_i [\theta_i w_i - < \theta_i > w_i])^2 P(\bar{\theta}|D) d^N\theta$$

$$(\sum_i [\theta_i w_i - < \theta_i > w_i])^2$$
$$= (\sum_i w_i [\theta_i - < \theta_i >])^2$$
$$= \sum_p \sum_q w_p w_q (\theta_p - < \theta_p >)(\theta_q - < \theta_q >))$$
$$= \sum_p \sum_q w_p w_q (\theta_p - < \theta_p >)(\theta_q - < \theta_q >))$$
$$\Delta f^2 = \int \sum_p \sum_q w_p w_q (\theta_p - < \theta_p >)(\theta_q - < \theta_q >) P(\bar{\theta}|D) d^N\theta$$
$$= \sum_p \sum_q w_p w_q (< \theta_p \theta_q > -2 < \theta_p >< \theta_q > + < \theta_p >< \theta_q >)$$
$$= \sum_p \sum_q w_p w_q (< \theta_p \theta_q > - < \theta_p >< \theta_q >)$$

which shows that to get the standard deviations on any linear function of the parameters it is enough to compute the parameter means $< \theta_i >$ and covariances $< \theta_p \theta_q >$.

In the case of a single parameter $w_p = \delta_{ip}$

$$\Delta f^2 = \Delta\theta_i^2 = < \theta_i^2 > - < \theta_i >^2$$

In the case of uncorrelated parameters $< \theta_p \theta_q > - < \theta_p >< \theta_q >= \delta_{pq}$

$$\Delta f^2 = \sum_i w_i^2 (< \theta_i^2 > - < \theta_i >^2)$$

In the general case, the off-diagonal elements give the correlation between different parameters, and can be either positive or negative, i.e. can either increase or decrease the error in the linear function.

Spectral Aspects

To get a total spectrum we average over regions of the total map for each energy range. The average total map is the sum of the average maps, but the errors on the total map must be computed by an appropiate sum over the total map corresponding to each sample. This can be done by defining the w_i based on the maps M_{ij}, and using the covariance matrix. The weights are defined by

$$w_i = \sum_{j \subset A} M_{ij} / \sum_{j \subset A} 1$$

where A is a mask defining the sky region for which the spectrum is required, and the denominator just normalizes to an average intensity over the region, as required for spectra in photons $\mathrm{cm}^{-2}\ \mathrm{sr}^{-1}\ \mathrm{s}^{-1}$.

Such spectra can be produced for any map regions, after the MCMC run (see below) is complete, so that the MCMC code does not need to use the maps explicitly, and only needs S_{ik}.

MCMC methods

Sampling from $P(\bar{\theta}|D)$ is the main computational problem, and it is proposed to use Markov Chain Monte Carlo (MCMC) methods. A good reference is Gilks et al. (1996), while Chen et al. (2000) is more advanced and technical but has useful sections. A clear and compact presentation of the basics is in Christensen et al. (2001). Usually we are interested in the posterior for particular parameters with the rest integrated out ('marginalization'). The Metropolis-Hastings MCMC algorithm is simple to implement and does not require any special sampling functions. Data-augmentation algorithms (van Dyk et al. 2001) should also be investigated, but these are considerably more sophisticated.

The *burn-in* phase is important in getting to the right region of the parameter space before starting to compute statistics. Lack of incomplete burn-in is manifested in a slow trend in the statistics with increasing samples, instead of random fluctuations about the converged values.

The posterior, average, standard deviation and covariance matrix are easy to obtain from the sampled set of $\bar{\theta}$.

Both uniform and Cauchy (= tan(π*uniform variate)) proposal distributions are implemented, for both 'local' (changing one parameter at a time) or 'global' (changing all parameters at once) schemes. In general it has been found that the local proposal distribution is more robust and less sensitive to the stepsize than the global scheme. This is presumably because it uses information per parameter so can more easily find an accepted point. For highly correlated basis functions (extreme case: two identical maps) however the local scheme needs a very small step since it can only move at an angle to the major axis and even then does not cover the full region; here the global scheme is much better, since it has a reasonable probability to sample along the major axis. The stepsize must then be correspondingly chosen to reflect the uncertainty in a single parameter.

In all cases the stepsizes for all parameters should be chosen by trial to give an acceptance ratio around 0.23 (NB find references on this topic) and the range 0.15-0.5 is recommended (Gilks et al. 1996, page 55) although the algorithm theoretically converges to the correct result for any acceptance ratio; the value quoted should optimize the rate of convergence. In has been found that a stepsize equal to the expected uncertainty on the parameter is a good trial choice. A possible way to automate this is to use the current estimate of the standard deviation to determine the step (Gilks et al. 1996), but still a starting step is required. The results are evaluated at intervals in the sampling (interval is user-defined) and it is important to check that there is no drift in the average values since this indicates lack of convergence and the result will then be influenced by the starting parameter values.

Chapter 4

Using *spimodfit*

From the user point of view, the data processing is done in 4 major steps, which all have their own options :

- Definition of the input observation data : on input, *spimodfit* expects the different files constituting a SPI observation (events rate, poitings, dead times, good times, energy bins and background models). The energy bins can be redefined and will be processed independently. It is also possible to select a subset of the detectors in the input data. The background models can be collected into one single model or limited to a maximum nulber (only the first ones will be loaded). In addition to these files, the Instrument response function files IRFs file has to be specified. Many IRFs files can be given to handle a time dependent response.
- Definition of the fitted parameters. *spimodfit* can fit extended emission ('images'), point sources ('sources') and background parameters. For each of these components, the time and detector variability should be given,as well as some other fitting or normalization parameters explained below.
- Fitting options, as the minimization routines and its initialization, MCMC options.
- Definition of the output files. spimodfitgenerate a FITS file table containing the results of the fit. By default this file contains the different skymodels (original map and fitted images) and an binary table containing the fitted parameters with associated configuration and estimated error bars. Optionally, it can also add an exposure map in this FITS file, create a background model for the observation, write residues of the model, light curves, apectra and an updated point source catalog in other files.

4.1 General parameters

First, let us begin with some general purpose parameters. These parameters do not influence the fitting process, they mainly are present for documentation purpose :

Parameter	: `debug`
Type	: integer
Range	:
Description	: Debugging flag, used during software development to dump some internal parameters to the screen. Leave the value to '0' (no debugging information) for normal use.

Parameter	: `title`
Type	: string
Range	:
Description	: This string will be written in the header of the output FITS data units (keyword `OBS_ID`). This record is only for information purpose and it is up to the user to give a good observation descriptor with less than 69 characters...

Parameter	: `output_idl`
Type	: integer
Range	: 0-1
Description	: If set to 1, print fitted parameters and associated errors on the screen. The values are separated by commas and enclose by saure brackets : `[val1,val2,...,valN]`. This is mainly intended to be copy-pasted from the screen (or, better, a log file) into an IDL code.

Parameter	: `rogroup`
Type	: string
Range	: Leave blank
Description	: The name of the read-only (i.e. input group). This parameter is needed by the `CommonPreparePARsStrings` routine, used to read the parameter file. Nevertheless, *spimodfit* doesn't support external input observation group and this parameter should be left blank.

4.2 Observation and IRF files

spimodfit fits an emission+bakground model w.r.t an SPI observation. For that purpose, the user needs to give the location of the different observation files, as well as the location of the IRF(s). Additionnal parameters allow to use a energy rebinned observation, select only a subset of detectors, filter the data counts or even simulate the detector events before fitting.

4.2.1 Loading and configuring the observation data

The observation data are loaded from 5 files :

- events count rate
- pointings
- energy bins definition
- dead time information
- good time intervals
- background index file

The user must specify the path to these files and some options that give the possibility to modify the observatoin structure and vlues.

4.2.1.1 Path to the observation files

The path to the observation files are given by the following parameters. By default, these files will be opened with read and write mode. In order to avoid data corruption if the programs exit abnormally, it is strongly recommended to make these files read-only or to gzipped them (compressed files can not be opened with write rights).

If the `simulate` parameter is set to 1 (see below), `counts_input_file` is the name of the output, simulated detector events. This file will thus be created by *spimodfit* and should not exist prior to the run.

Parameter	: `counts_input_file`
Type	: string
Range	:
Description	: Location of the detector events table, as existing in a SPI observation. The index of the binary table can be omitted, e.g `evts_det_spec.fits`, in which case *spimodfit* will look for the first binary table named `SPI.-OBS.-DSP`

Parameter	: `pointing_input_file`
Type	: string
Range	:
Description	: Location of the pointing definition file, as existing in a SPI observation. The index of the binary table can be omitted, e.g `pointings.fits`, in which case *spimodfit* will look for the first binary table named `SPI.-OBS.-PNT`

Parameter	: `ebounds_input_file`
Type	: string
Range	:
Description	: Location of the energy binning definition file, as existing in a SPI observation. The index of the binary table can be omitted, e.g `energy_boundaries.fits`, in which case *spimodfit* will look for the first binary table named `SPI.-EBDS-SET`

Parameter	: `deadtime-dol`
Type	: string
Range	:
Description	: Location of the dead time definition file, as existing in a SPI observation. The index of the binary table can be omitted, e.g `dead_time.fits`. If not specified, *spimodfit* will look for the first binary table named `SPI.-OBS.-DTI`

Parameter	: `gti-dol`
Type	: string
Range	:
Description	: Location of the good time intervals definition file, as existing in a SPI observation. The index of the binary table can be omitted, e.g `gti.fits`, in which case *spimodfit* will look for the first binary table named `SPI.-OBS.-GTI`

Parameter	: `background_input_file`
Type	:
Range	: Location of the background model index file, as produced by, e.g, `spiorthomodel` or `spiback`. The index of the binary table can be omitted, e.g `gti.fits`, in which case *spimodfit* will look for the first binary table with the group name `SPI.-BMOD-DSP-IDX`
Description	:

4.2.1.2 Background models selection and combination

The background model index file pointed by `background_input_file` can contain a lot of different components. The following paraeters allow the user to select only a subset the component (the first ones) and/or combine the loaded components into a single one.

Parameter	: `n_background_loaded`
Type	: integer
Range	: ≥ 1
Description	: This parameter sets the maximum number of loaded background components (rejecting the last ones in the index). This is mainly intended for use in conjunction with `spiorthomodel`. Actually, by default this software records the different background components in decreasing order of pertinence. This parameter allows then to run the model fitting with different refinements of the background model.

Parameter	: `collect_background_models`
Type	: integer
Range	: 0-1
Description	: If set to 1, all the backgournd components are collected into one, and the process continues as if there is only one component in the background model.

4.2.1.3 Energy rebinning of the input data

The observation data can be selected and rebinnned in energy. This is done by specifying the first and last bins, and the number of bins per rebinnined energy. These bin numbers refer to the bins sequence of the original data.

Parameter	: `first_energy_bin`
Type	: integer
Range	: ≥ 1
Description	: First selected bin, as indexed in the original observation data, beginning at 1.

Parameter	: `last_energy_bin`
Type	: integer
Range	: ≥ 1
Description	: Last selected bin, as indexed in the original observation data, beginning at 1

Parameter	: `m_energy_rebin`
Type	: integer
Range	: ≥ 1
Description	: Number of original bins per rebinned energy.

4.2.1.4 Detector selection of the input data

The input data can also be filtered to select only the events from a given subset of (pseudo)detectors. The selected detectoirs are given with the `detector_selection` parameter. There is usually no need to reject dead detectors, since they are automatically taken into account and their parameters not fitted. This paremetr is more intened to select single and/or multiple events in a given observation.

Parameter	: `detector_selection`
Type	: list of integer ranges
Range	:
Description	: Comma separated list of selected detectors or detector ranges. For example, the list '00, 01, 03-18' select events from dectors 00, 01 and 03 to 18.

4.2.2 Simulation of detector events

Instead of fitting the model directly from an existing data set, *spimodfit* can generate simulated detector events (Monte Carlo sampling), based on the provided emission and background models and the observation characteristics (pointings, dead times, etc.). Once the data have been simulated, the fitting process continues as usual.

At the end of the programm, a count rate FITS file is created (with the name given by `counts_input_file`, which then acts as an output parameter instead of an input one), and the simulated counts recorded in it.

Data simulation is switched on by the `simulate` parameter.

Parameter	: `simulate`
Type	: integer
Range	: 0,1
Description	: If set to 1, the data are first simulated (Monte Carlo sampling) and the resulting counts rate recorded in `counts_input_file`. The fitting process is then performed as usual.

4.2.3 Rejection of bad data counts and background smoothing

In some early version of standard data processing, some data counts could have been set to spurious, negative values. The detector events with counts number lower than a user-defined limit (this parameter) are then set to 0, as well as the corresponding term in the convolution matrix. This behaviour appears now deprecated and it is recommended to keep the threshold to its default value (-1) so that negative counts shall be filtered (but this should not happen anymore...).

Parameter	: `data_filter_counts`
Type	: integer
Range	:
Description	: All data counts lower than this threshold will be filtered out. [DEPRECATED]

One of the way to regularize the background model is to generate a model from the count rate from an "off" observation and use it directly as a background model for an "on" observation. This method can nevertheless exhibit a strong variability of the model and introduces unexpected short term variation. To avoid this problem, the background model can be smoothed (using a sliding mean) over a given number of energy bins, set by `background_smooth`. If set to one, no smoothing is performed. `background_smooth` must be odd.

Parameter	: `background_smooth`
Type	: integer
Range	: ≥ 1
Description	: Number of energy bins used for background smoothing, must be odd.

4.2.4 Loading the IRF files

spimodfit needs to load Instrument Response Functions (IRFs) in order to calculate the expected detector count rates from a given sky distribution. The program can handle more than 1 IRF, to take into account changes in the instrumental response, such as the loss of detectors (which is not exactly equivalent to set the corresponding livetimes to 0). If no sky nor point source emission is fitted (e.g. for background fitting), then no IRF is required. The number of loaded IRF is set with `n_irf` and their location by `irf_input_file_##`, where ## stands for the IRF number (from 01 to `n_irf`).

Parameter	: `n_irf`
Type	: integer
Range	: ≥ 0
Description	: Number of loaded time-dependent IRFs. Can be set to 0 when no emission model is given.

Parameter	: `irf_input_file_##`
Type	: string
Range	:
Description	: Location of the IRF grouping file niumber ##. The indication of the extension number is optional.

Two additionnal parameters are also used when loading the IRFs.

If only one enegy bin is fitted and all the 'gamma correction' for the sky sources are identical (see the corresponding section), the instrument response can be pre-calculated (for a given

observation), thus saving memory and time. If the `precalculate_irf` parameter is set to 1, then the response is pre-calculated (if possible). If the precalculation is not possible, a warning message is issued and the algorithm continues using 'on the fly' response calculation.

Parameter	: `precalculate_irf`
Type	: integer
Range	: 0-1
Description	: Pre-calculate instrument response if possible. The precalculation is only possible if only one energy bin is fitted and the gamma (i.e. source spectral index) corrections are identical.

In order to calulate the instrument response, the IRF has to be integrated in the energy domain. This is done with a 4 points interpolation on a logarithmic energy sampling. The number of integration interval in $[E_{min} - E_{max}]$ is then given by $(\log(E_{max}) - \log(E_{min}))/lstep$, $lstsep$ is the logarithmic step as given by the parameter `log_integration_step_irf`. It is recommended to keep this parameter to its default value of 0.03. A zero value causes the integration to be performed on a single interval $(E_{min} - E_{max})$.

Parameter	: `log_integration_step_irf`
Type	: real
Range	: ≥ 0
Description	: Logarithmic (base 10) integration interval of the IRF.

4.3 Definition of the fitted components

spimodfit load and fit 3 types of components : images (i.e extendend sky emissions), point sources and background models. For each of these components the parameters constraints, variability order, etc should be set in the parameter file.

4.3.1 Image components

The provided sky emission models has to conform to the OSA standard for images, that is with one image index refering to a FITS image extension.

4.3.1.1 Number and location of sky models

The number of images to load is given by the `n_image_parameters` parameter :

Parameter	: `n_image_parameters`
Type	: integer
Range	: ≥ 0
Description	: Number of sky emission models

spimodfit requires the user to provide as many image indexes as sky emission models. An image index is a FITS data unit containing the location of 1 or more regular image extensions. This is the standard way of specifying sky models in ISDC softwares. If more than one image are listed in the index file, then the corresponding sky models are assumed to be energy dependant and *spimodfit* will rely on the image keyword E_MIN and E_MAX to select the correct models to be convolved for a given fitted energy range. Usually, only 1 image is defined in the indexes and, in this case, the same sky model is used for all energy ranges (regardless of the values of E_MIN and E_MAX). The locations of the index files are specified with the parameters `image-idx_##`, where ## stands for the number (from 01 to `n_image_parameters`) of the model component :

Parameter	: `image-idx_##`
Type	: string
Range	:
Description	: Location of the index for the image component ##. The extension number doesn't need to be precised, *spimodfit* looking for the extensions named `SPI.-SKY.-IMA-IDX`, but their should not be more than one index table per FITS file.

4.3.1.2 Sky models units and energy-dependent rescaling

The unit of the input maps is assumed to be in $ph \cdot s^{-1} \cdot cm^{-2} \cdot sr^{-1} \cdot keV^{-1}$, while the output maps for each energy will be rescaled to $ph \cdot s^{-1} \cdot cm^{-2} \cdot sr^{-1}$, that is, integrated over energy. The spectral shape of the sky emission can be precised through the three following parameters and its integral is used to ponderate the images for each energy bin. More precisely, at each energy, the fitted map is multiplied by $Ef = \int_{E_{min}}^{E^{max}} f_s(E)dE$, where E_{min} and E^{max} are the boundaries of the energy bin being fitted and f_s is the spectral shape. f_s has to be a valid arithmetic expression. *spimodfit* uses the `FastMathParser` (version 1.09) to parse any expression involving the basic operators (+,-,/,*,^), conditionnal relations (<=, >=, ! =, ==, <, >, and, or) and single or multiple arguments functions :

- trigonometric function : sin, cos, tan, asin, acos, atan
- hyperbolic functions : sinh, cosh, tanh, asinh, acosh, atanh
- logarithm functions : log2, log10, log (alias for log10), ln
- miscellaneous functions : exp, sqrt, sign (gives the sign of an expression), rint (nearest integer), abs, if.
- variable number of arguments functions : sum (the sum of the arguments), avg (mean value), min (min of the arguments), max (max of the arguments.

The two numerical constants '_pi' and '_e' are also supported. When using these functions, the only non-fixed parameter should be the energy in keV, refered as the variable 'E', e.g : 1.0/(sqrt(2*_pi)*3.0)*exp(((E-511)/3.0)^2)*E^(-1.2).

These functions should be sufficient to define any relevant spectral shape... Nevertheless, to ease the formulation and to conform to the XSPEC standard, 7 functions 'shortcuts' have been defined :

- powerlaw : $A * E^{-\gamma}$, requiring the additional parameters γ, A.
- cutoff : $A * E^{-\gamma} * \exp(-E/E_{cut})$, requiring the additional parameters γ, E_{cut}, A.
- bknpower : $A*E^{-\gamma_1}$ if $E < E_{break}$ and $A*E_{break}^{\gamma_2-\gamma_1}*E^{-\gamma_2}$ otherwise, requiring the additional parameters $\gamma_1, E_{break}, \gamma_2, A$
- gaussian : $A/(\sqrt{2\pi}\sigma) * \exp(-0.5 * ((E - E_p)/\sigma)^2)$, requiring the additional parameters E_p, σ, A.
- constant : $A\ \forall E$, requiring the additional parameter A
- highecut : 1.0 if $E \leq E_{cut}$ and $\exp(-(E-E_{cut})/E_{fold})$ ortherwise, requiring the additional parameters E_{cut}, E_{fold}.
- wabs : should be the absorbtion by a column density N_h of hydrogen. In *spimodfit* , This function returns inconditionnaly 1 and is only implemented for compatibility with most of the XSPEC formulations. It requires an additional (not used) parameter N_h

When using these functions, the additonal parameters (that is all except the energy) are specified within a separate list. The order of the parameters in the list should match the order of the functions in the formula and, for each function, the order of parameters as given in the above description of the function. For example, the above spectral shape (exp(((E-511)/3.0)^2)*E^(-1.2)) can also be entered as "gaussian*powerlaw" with the additional parameter list "511, 3.0, 1.0, 1.2, 1.0", the three first values being used by "gaussian" and the two last ones by "powerlaw".

The formula is entered with the `image_energy_model_##` parameter and any additional, required values with `image_energy_model_pars_num_##` and `image_energy_model_pars_##`.

Parameter	: `image_energy_model_##`
Type	: string
Range	:
Description	: Spectrum shape of image component ##. This should be a valid mathematical expression, optionnaly involving XSPEC energy models, whose parameters are given by the following parameters. See above for details

Parameter	: `image_energy_model_pars_num_##`
Type	: integer
Range	: ≥ 0
Description	: Number of parameters required to evaluate the XSPEC functions defined in the spectrum model for image component ## .

Parameter	: `image_energy_model_pars_##`
Type	: vector of reals
Range	:
Description	: Vector of parameters, as required by the spectrum model for image component ##.

Gamma correction for IRF The IRF interpolation within an energy bin assumes that the component spectrum follows a powerlaw ($F \propto E^{-\gamma}$). The powerlaw exponent (γ) can be set via `image_IRF_gamma_cor_##`. For relatively narrow energy bins, this parameter is expected to have only little influence on the output result.

Parameter	: `image_IRF_gamma_cor_##`
Type	: real
Range	: ≥ 0
Description	: Gamma correction of the IRF within an energy bin, for image component ## .

4.3.1.3 Definition of detector and time intervals

As for other fitted components (sources and background models), the user has to specified the detector and time ranges to be (optionnaly) fitted. This is done via the `image_det_rges_##`, `image_var_coef_##` and `image_var_order_##` parameters.

Detector selection The argument of `image_det_rges_##` is a list of individual or ranges of detectors. For example, `'00-03,04,05-18'` will define three 'virtual' detector blocks : the first one with detectors 01 to 03, the second one containing the single detector 04 and the third one with detectors 05 to 18. These 3 blocks will be fitted independantly. *Please note that it is not possible to define block with non-following detector numbers.*

Parameter	: `image_det_rges_##`
Type	: list in integer ranges
Range	:
Description	: Definition of the detector 'blocks'. Each block is fitted independently

Time selection The observation can be divided into time intervals that are fitted independently. `image_var_coef` allows to define an arbitrary time variability and, therefore, more flexibility in the time dependance of the image components. The argument of `image_var_coef` is a string composed of comma separated fields, each of one defining additionnal time nodes (i.e. boundaries of the time intervals). Each field define the variability either by periodicity or nodes' values. The unit of the variability is either in days or in pointings. More precisely the format of a field consists of numbers, separated by blank characters, followed by 2 characters specifying

the variability unit (d=dyas, p=pointings) and the variability type (i=increments, n=nodes). For example, the string '1000 2000 d n, 3.0 d i' will define 1 time node every 3 days, plus 2 specific time nodes at 1000 and 2000 ISDC Julian days.

Between nodes, the time varaibility is defined as a piecewise function of parametrized regularity (i.e derivability) order. Thus, a zero order variability means a constant term for each time interval Order 1 ensures linear variation so that the function is continuous. Order 3 corresponds to the well known cubic spline. Possible varaibility orders are form 0 to 5, odd numbers (+0) are *strongly* recommended. Note that this mechanism requires additiionnal parameters (as many as the desired order) for each set of contiguous intervals.**As a consequence, fitting an order 1 or more time variability usually fails because of some over-parametrized time intervals. This could be considered as a bug ...**.

Parameter	: `image_var_coef_##`
Type	: list of real vectors + specifiers
Range	: ≥ 0 + "p/d" + "i/n"
Description	: Specifies the nodes of the time variability for image component ##. The format of the argument is detailed in the above text.

Parameter	: `image_var_order_##`
Type	: integer
Range	: 0..5
Description	: Regularity order of the time variability. Recommended to be 0 or odd. **A variability order greater than 0 usually induces an unstable fitting process. Use with care...**

4.3.1.4 Default parameters values and fitting flags

The optimization process consists of finding a linear combination of components (images + point sources + background models) that minimizes a given function (either least squares or maximum likelihood optimization). In *spimodfit* , the fitted parmeters are unitless, scaling coefficients of the different components, after having applied energy-dependent rescaling and integration (for the emission models).

In some cases, it may be useful to define a "fixed" (i.e. not fitted...) sky emission. This can be done with `image_parameters_fit_##`. If set to 0, the parameters are held fixed, set to their default values (as defined by `image_parameters_##`).

Parameter	: `image_parameters_fit_##`
Type	: integer
Range	: 0,1
Description	: Fitting flag for the parameters of image component ##. If 0, the parameters are not fitted, set to their defaults values (as defined by `image_parameters_##`

When using the MCMC procedure for fitting and error estimation, the user has to define the parameter step to be multiplied by the uniform of Cauchy distribution. The step value is defined with `image_parameters_step_##`.

Parameter	: `image_parameters_step_##`
Type	: real
Range	: > 0
Description	: Parameter step, to be multiplied by the uniform or Cauchy distribution, used during the MCMC fitting process. Relative to the image component ##.

The default parameter values and boundary constraints can be set either globally for all fitted parameters of a given image component, or from an initialization file.

Global parameters initialization In this paragraph, it is assumed that no initialization file is used (see below). Otherwise, the loaded values from the file override the ones defined in the parameters file.

Unless using an unconstrained least square fit, the fitting processes require to start from default parameters. They can be set for an image component via `image_parameters_##`.

Parameter	: `image_parameters_##`
Type	: real
Range	:
Description	: Default fitted parameter value (rescaling factor) for image component ##. This value is overridden in case of initialization by an external file or an unconstrained least squares fit.

Moreover, the fitting process handles boundary constraints on parameters. These are set globally for all parameters of a given image components by using `image_parameters_min_##` and `image_parameters_max_##`.

Parameter	: `image_parameters_min_##`
Type	: real
Range	:
Description	: Low boundary constraint of image component ## parameters. It is overridden in case of the use of an initialization file.

Parameter	: `image_parameters_max_##`
Type	: real
Range	:
Description	: High boundary constraint of image component ## parameters. It is overridden in case of the use of an initialization file.

Initialization from an external file Optionnaly, the image parameters (for all components) can be initialize from an output file (FITS format). This possinility is mainly intended to start a new fit from the results of a previous one. Consequently, the format of the FITS file should follow a precise format. Moreover, the number of image parameters defined in the file must match the number of parameters deduced from the number of sky models, the detectors blocks and the time variability.

First, *spimodfit* looks for a data extension defining the energy bins (named and conforming to the `SPI.-EBDS-SET` template). One of the defined energy bin shall correspond to the current fitted energy interval. The index of the corresponding bin is then used to move to the corresponding parameters data unit (named `SPI.-DFIT-RES_##`, where ## stands for the index of the energy bin). This data unit has to contain, at least, 4 columns, with one row per parameter :

- **PAR_TYPE :** This string column specifies the 'type' of the parameters. If the string contains 'Sky', then it is considered as an image parameter. Similarly, 'Point' refer to a point source and 'Background' to a background parameter.
- **MIN_VALUES :** Low boundary constraint of the parameter.
- **MAX_VALUES :** High boundary constraint of the parameter.
- **FIT_VALUES_ML :** As an output of *spimodfit* , this column correspond to the fitted values through the maximum likelihood or least squares method (not MCMC). As an initialization file, this column is used to set the defaults values of the parameters.

The initialization file for images is loaded if `image_init_from_file` is set to 1. The name of the corresponding file is given by `image_init_file`.

Parameter	: `image_init_from_file`
Type	: integer
Range	: 0,1
Description	: If set to 1, an initialization file is loaded to set the defaults values, low and high boundary constraints of all image parameters. The name of the initialization file is given by `image_init_file`.

Parameter	: `image_init_file`
Type	: string
Range	:
Description	: Location and name of the initialization file. This file, if `image_init_from_file` is set to 1, is loaded to get the defaults values, low and high boundary constraints of all image parameters.

4.3.1.5 Configuration of the output sky models

The fitted sky models ('images') are stored and recorded in the output fits file in Galactic or celestial (J2000) coordinates. Moreover, in this coordinate system, the images can be limited to a rectangular area (every pixel lying outside this area will be set to zero). Additionnally, the size of the sky pixels (as used in the convolution process and image recording) has to be defined.

The coordinate system of the output map is set with `skymap_system` ('G' for galactic, 'C' for celestial).

Parameter	: `skymap_system`
Type	: string
Range	: 'G','C'
Description	: Set the coordinate system of the output sky models. 'G' for galactic, 'C' for celestial (J2000)

The boundaries of the fitted sky area is set with the `chi_0`, `chi_1` (min and max values in longitude) and the `psi_0`, `psi_1` (min and max values in latitude). `d_chi` and `d_psi` set the size of the sky pixel.

Parameter	: `chi_0`
Type	: real
Range	:
Description	: Minimum longitude of the sky model area, in degrees

Parameter	: `chi_1`
Type	: real
Range	:
Description	: Maximum longitude of the sky model area, in degrees

Parameter	: `d_chi`
Type	: real
Range	:
Description	: Pixel size along the longitude, in degrees

Parameter	: `psi_0`
Type	: real
Range	:
Description	: Minimum longitude of the sky model area, in degrees

Parameter	: `psi_1`
Type	: real
Range	:
Description	: Maximum longitude of the sky model area, in degrees

Parameter	: `d_psi`
Type	: real
Range	:
Description	: Pixel size along the latitude, in degrees

4.3.2 Point sources components

The definition of point sources component is primarly performed through the use of a source catalog, conforming to the `SPI.-SRCL-CAT` (SPI source catalogue) or `GNRL-REFR-CAT` (general reference catalogue). The location of the input catalogue is given with `source-cat-dol`. If its value is left empty, then no point source is loaded.

Parameter	: `source-cat-dol`
Type	: string
Range	:
Description	: Path and filename of the point sources catalogue, conforming to the `SPI.-SRCL-CAT` or `GNRL-REFR-CAT` template format. No point source is loaded if the field is left empty.

The sources' catalog contains a ineteger column named `SEL_FLAG`. Only sources with `SEL_FLAG` set to a non-zero value will be loaded and used for fitting in *spimodfit* .

4.3.3 Source flux

As for the images components, the fitted parameters for point sources are unitless scaling factors. It is therefore necessary to set a reference flux for each source, in $ph \cdot s^{-1} \cdot cm^{-2} \cdot keV^{-1}$. In the same step, an energy dependence of the flux can be set (i.e. the definition of a reference spectrum). There are three different ways to define the reference source spectrum : the same pre-defined spectral law for all sources (from the parameter file) or individually for each source (from the catalogue) or a recorded spectrum per source (from the catalogue).

Index of *spimodfit* parameters

background_ input_ file, 19
background_ smooth, 22

collect_ background_ models, 20
counts_ input_ file, 18

data_ filter_ counts, 21
deadtime-dol, 19
debug, 17
detector_ selection, 21

ebounds_ input_ file, 19

first_ energy_ bin, 20

gti-dol, 19

image-idx_ ##, 24
image_ det_ rges_ ##, 26
image_ energy_ model_ ##, 25
image_ energy_ model_ pars_ ##, 26
image_ energy_ model_ pars_ num_ ##, 25
image_ init_ file, 29
image_ init_ from_ file, 29
image_ IRF_ gamma_ cor_ ##, 26
image_ parameters_ ##, 28
image_ parameters_ fit_ ##, 27
image_ parameters_ max_ ##, 28
image_ parameters_ min_ ##, 28
image_ parameters_ step_ ##, 28
image_ var_ coef_ ##, 27
image_ var_ order_ ##, 27
irf_ input_ file_ ##, 22

last_ energy_ bin, 20
log_ integration_ step_ irf, 23

m_ energy_ rebin, 20

n_ background_ loaded, 20
n_ image_ parameters, 23
n_ irf, 22

output_ idl, 17

pointing_ input_ file, 18
precalculate_ irf, 23

rogroup, 17

skymap_ system, 30

title, 17